BOTTEGA
IT minds

Program szkolenia:

React - głębokie zanurzenie

Informacje:

Nazwa:	**React - głębokie zanurzenie**
Kod:	**react-deep**
Kategoria:	React
Odbiorcy:	developerzy, architekci
Czas trwania:	4 dni
Forma:	warsztaty

Otwórz nowe horyzonty swojemu zespołowi dzięki wejściu wgłąb Reacta, dotykając zagadnień, po których tutoriale tylko prześlizgują nie dając możliwości ich zasadniczego zrozumienia.

Głębsze spojrzenie na Reacta w Twoim zespole umożliwi rozwijanie projektu w czytelny architektonicznie sposób, a poznane metody pisania kodu zwiększą jego niezawodność i utrzymywalność.

Wprowadź ekspercką wiedzę do swojego zespołu w zakresie architektury komponentowej oraz wpływu konstrukcji Reacta na Wasze decyzje architektoniczne. Poznaj narzędzia wspierające utrzymanie dobrej architektury aplikacji. Z poziomu, w którym projekt już na starcie stał się zagmatwany i rozwijany po omacku, wejdź na poziom, gdzie świadomie wydzielasz konkretne odpowiedzialności, stosujesz krytyczne zasady przepływu danych i zarządzania stanem, a projekt stanie się prosty dzięki odpowiednim wzorcom tworzenia API komponentów.

Programiści Reacta z wieloletnim doświadczeniem, poznając prezentowane zagadnienia, najczęściej odpowiadają:
„W ogóle nie myślałem o tym w ten sposób, a jest to ultraciekawe i bardzo ważne. Wychodzi na to, że nie znałem Reacta”.

Szkolenie jest prowadzone w formie warsztatów, podczas których pracujemy na realnych problemach, krok po kroku odkrywając problematykę zagadnienia i możliwe kierunki rozwiązania. Jest oparte na dwóch filarach – filarze faktów i nauki oraz filarze opinii i osobistego doświadczenia trenera. Pierwszym filarem jest bliższe przyjrzenie się Reactowi od strony, której zazwyczaj nie dotykamy, pisząc aplikacje, a której zgłębienie pozwoli inaczej spojrzeć na pisany kod, istniejące ograniczenia oraz poznać niewykorzystane dotąd możliwości Reacta. Drugim filarem są praktyczne ćwiczenia organizacji struktury komponentów i odpowiedzialności w kodzie, czyli temat po prostu unikany na wielu szkoleniach, a który to właśnie ma największy wpływ na to, co się dzieje w Waszych projektach.

Techniki i narzędzia wykorzystywane podczas szkolenia:

- Storybook
- Jest/Mocha
- React Testing Library
- TypeScript
- typy generyczne w zakresie podstawowym
- Zod
- json-server
- repozytorium GitHub

BO·TT·EGA
IT minds

Czym to szkolenie nie jest?

- podstawowym kursem Reacta
- przeglądem API Reacta
- przeglądem dostępnych i najświeższych technologii
- wprowadzeniem do TypeScript

Proponowany przebieg szkolenia:

- Każdy uczestnik wypełnia ankietę z ćwiczeniami do samodzielnego rozwiązania – pozwoli to na dostosowanie poziomu szkolenia do konkretnego zespołu deweloperskiego.
- Zespół dostarcza trenerowi fragment kodu napisanego w React, nad którym obecnie pracuje.
- Trener spotyka się z zespołem on-line przed szkoleniem, aby omówić największe wątpliwości zespołu co do rozwijanego przez siebie projektu.
- Na podstawie zebranych informacji trener dostosowuje materiał szkolenia do realiów codziennej pracy zespołu.
- W części warsztatowej uczestnicy pracują w większości na przygotowanym wcześniej kodzie, w parach, na zasadzie pair programmingu.
- Końcowy etap szkolenia polega na wybraniu dowolnego elementu w kodzie zespołu i próbie usprawnienia go przy zastosowaniu wiedzy zdobytej na szkoleniu.
- Może to być na przykład wydzielenie dodatkowych komponentów, zmiana API, naprawa nadużyć Reacta w kodzie, czy przemodelowanie stanu i przepływu danych, albo reorganizacja kodu pod kątem bardziej praktycznych testów.

Zalety szkolenia:

- Będziesz podejmować bardziej świadome decyzje projektowe i optymalizacyjne.
- Unikniesz wielu potencjalnych bugów związanych z nieodpowiednim modelowaniem i wykorzystaniem stanu oraz interakcji.
- Będziesz wiedzieć, jak tworzyć kod łatwy w utrzymaniu i łatwy do przyswojenia dla kolegów z zespołu.
- Zrozumiesz w końcu, czym jest komponent funkcyjny i czym zasadniczo różni się od klasowego.
- Będziesz stosować techniki z TDD, które nie są bezpośrednio dostępne podczas implementowania komponentów UI.
- Będziesz lepiej rozumieć możliwości i ograniczenia przy tworzeniu komponentów i łączeniu ich w większe części dzięki poznaniu całego cyklu życia Reacta – od składni JSX do samej mutacji DOM.
- Będziesz się lepiej orientować podczas debugowania komponentów reactowych, a Virtual DOM zobaczysz z bliska „gołym okiem”, poznasz go z innej strony niż z ogólnikowych haseł z Internetu.
- Nauczysz się lepiej modelować stan aplikacji, prawidłowo odwzorowując fakty i unikając sytuacji niemożliwych lub wieloznacznych – dzięki zastosowaniu tzw. discriminated union w TypeScript.
- Poznasz prosty sposób na rozdzielenie wyglądu, logiki interakcji, asynchronicznego I/O, a nawet integracji z konkretnym backendem dzięki wprowadzeniu wzorców czystych komponentów, layoutu, hooków zarządzających, klienta I/O, adaptera typów serwerowych i aplikacyjnych.
- Wypełnisz największą lukę w swoim idealnie otypowanym projekcie poprzez wprowadzenie prostych i niesamowicie wygodnych tzw. runtime parserów typów w TypeScript.
- Będziesz w stanie przeorganizować swój projekt tak, aby testowanie rzeczy ważnych było łatwiejsze i przyjemniejsze podczas pisania testów, bez konieczności mockowania modułów albo wprowadzaniu specjalnych hacków w kodzie, czy też reprodukowania nieistotnego stanu tylko po to, żeby komponent w ogóle dał się przetestować.

BOTTEGA
IT minds

Szczegółowy program:

1. TypeScript – środowisko występowania typów, typowanie strukturalne, inferencja

2. Renderowanie rozebrane na czynniki pierwsze

2.1. Składnia JSX, drzewa pełne i płaskie

2.2. Rekoncyliacja, problem złożoności czasowej, heurystyki porównawcze, pełne wyjaśnienie key attribute

2.3. Faza commit

2.4. Re-render

2.5. Komponent vs. element vs. instancja komponentu

3. Komponenty funkcyjne

3.1. Omówienie zasadniczych różnic względem komponentów klasowych, wyjaśnienie genezy krytycznej zasady normalizacji stanu

3.2. Domknięcia w JS i błędne postrzeganie przepływu danych w komponencie funkcyjnym

3.3. Sposób pozyskiwania stanu w funkcji (useState) -– wyjaśnienie powodu istnienia tzw. rules of hooks

3.4. UseEffect – omówienie różnic względem klasycznych metod cyklu życia, omówienie ograniczenia w przekazywaniu zależności

4. Rekurencja i mylna intuicja

4.1. Praktyczne ćwiczenie, które pozwoli na zrozumienie problemu granic odpowiedzialności między komponentami w jej najtrudniejszym wydaniu – granicą między samym sobą.

4.1.1. Lepiej zrozumiesz współpracę funkcji renderującej oraz mechanizmu renderującego

4.1.2. Zrozumiesz zasadniczą różnicę między klasyczną rekurencją a rekurencyjnym renderowaniem.

5. Architektura

5.1. Przegląd metod zarządzania stanem (redux, useState, useReducer, immer, …)

5.2. Make Impossible State Impossible – modelowanie prawdy w TypeScript

5.3. Service hooks – separacja obsługi interakcji i widoku

BOTTEGA IT minds

5.4. Wzorzec klienta – separacja I/O

5.5. Model adaptera – wspieranie różnych interfejsów na backendzie

5.6. Zabezpieczanie typów w runtime

5.7. Izolacja domen – czyli kto z kim powinien mieć zależności

5.8. Komponentowe „dziel i rządź” – bo przecież nikt nie lubi spaghetti

5.9. Storybook – czyli TDD które jest do zrobienia

5.10. React Testing Library – testowanie integracyjne komponentów

5.11. Testowanie service hooks

6. Praca na kodzie własnym wskazanym przez zespół